DESIGN AND DEVELOPMENT OF EMBEDDED SYSTEM AND GUI FOR PULSATING SENSOR BASED INSTRUMENTATION

AN INDUSTRIAL INTERNSHIP REPORT

submitted by

JENCY RUFINA A

(23BEL1037)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRICAL AND COMPUTER SCIENCE ENGINEERING

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

OCTOBER 2025

DECLARATION BY THE CANDIDATE

I hereby declare that the mini project report entitled "**DESIGN AND DEVELOPMENT OF EMBEDDED SYSTEM AND GUI FOR PULSATING SENSOR BASED INSTRUMENTATION"** submitted by me to Vellore Institute of Technology - Chennai Campus, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Electrical and Computer Science Engineering** is a record of bonafide industrial training undertaken by me under the supervision of **Dr. Ramesh Sanga**. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place:

Date: Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF ELECTRICAL ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the in-plant training report entitled "**DESIGN AND DEVELOPMENT OF EMBEDDED SYSTEM AND GUI FOR PULSATING SENSOR BASED INSTRUMENTATION"** submitted by **JENCY RUFINA A(23BEL1037)** to Vellore Institute of Technology - Chennai Campus, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Electrical and Computer Science Engineering** is a record of bonafide in-plant training undertaken by him/her under my supervision. The training fulfills the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Date: **Signature of the HoD**

Examiner (s) Signature

1. **2.**

परमाणु ऊर्जा विभाग
Department of Atomic Energy
भारत सरकार
GOVERNMENT OF INDIA
सत्यमेव जयते
इन्दिरा गांधी परमाणु अनुसंधान केन्द्र
Indira Gandhi Centre for Atomic Research
कल्पाक्कम तमिलनाडु भारत
KALPAKKAM. TAMIL NADU. INDIA

TECHNOLOGIES FOR
NEW INDIA@75
आज़ादी का अमृत महोत्सव

M.SIVARAMAKRISHNA
SCIENTIFIC OFFICER 'H'
HEAD, SECURITY&INNOVATIVE SENSORS DIVISION
ELECTRONICS AND INSTRUMENTATION GROUP,
IGCAR, KALPAKKAM,
CHENGALPATTU Dt , PIN - 603102

email : sivarama@igcar.gov.in
Phone : 044 27480500 -22491(O)
044 27480500 -89357(R)
044 27482334(R), 9444901291(O)
9629538203(O –Whatsapp)
044-27480062 (O-BSNL)
igcarsiva@gmail.com
igcarsivaa@gmail.com

INTERNSHIP CERTIFICATE

I certify that Ms. Jency Rufina A, 23BEL1037 studying B.Tech ECSE (Batch 2023-2027), in Vellore Institute of Technology, Chennai has successfully completed Project work on "Design and Development of Embedded System and GUI for Pulsating Sensor based Instrumentation", in our division of Indira Gandhi Centre for Atomic Research, under the guidance of Dr. Ramesh Sanga, SO/E, during 28th May 2025 to 8th July 2025. During the period of project work, she was sincere, dedicated and her behavior was satisfactory.

Date of issue: 11/08/2025

[illegible]
एम. शिवराम कृष्णा / M. SIVARAMA KRISHNA
प्रधान, सुरक्षा एवं अभिनव संवेदक प्रभाग
Head, Security & Innovative Sensors Division
आईआरएसजी / IRSG
इंदिरा गांधी परमाणु अनुसंधान केंद्र
Indira Gandhi Centre for Atomic Research
कल्पाक्कम / Kalpakkam - 603 102

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Dr. Ramesh Sanga**, ISS/SISD/EIG/IGCAR, for his invaluable guidance and mentorship throughout the course of this mini project. His deep expertise in the field of Electronics and Instrumentation, coupled with his technical insight, meticulous attention to detail, and unwavering dedication, has played a crucial role in the successful completion of this work.

I am also deeply thankful to **Shri M. Sivaramakrishna**, Head, SISD/EIG/IGCAR, for his kind support and encouragement during the course of my project.

Furthermore, I extend my heartfelt thanks to **VIT, Chennai**, for granting me this opportunity. I am truly grateful for the education, resources, and support I have received, which have significantly influenced my academic and scientific growth.

Thank you.

Place: Chennai

(Jency Rufina A)

Date:

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1.	**Overview of IGCAR**	**1**
	1.1 IGCAR	1
	1.2 Publications by Dr. Ramesh Sanga ISS/SISD/EIG	3
2.	**Introduction**	**4**
	2.1 Objective	5
	2.2 Spirometer	5
	2.3 GUI and PyQt5 Overview	6
	2.4 Communication Protocols Used	6
3.	**Design and Development of Embedded Systems for Pulsating sensor**	**8**
	3.1 Embedded Hardware Design	8
	3.2 Timers in Arduino Mega 2560	9
	3.3 DS3231 RTC	13
	3.4 TFT LCD Shield	14
	3.5 Micro SD Card Module	15
4.	**Design and Development of Embedded Systems for Spirometer**	**17**
	4.1 Venturi tube	18
	4.2 ADP810 Sensor	19
	4.3 Arduino UNO and Methodology	20
5.	**Design and Development of GUI for Embedded Systems**	**21**
6.	**Testing of the Developed Embedded Systems**	**26**
7.	**Conclusion and Future Work**	**34**
8.	**APPENDIX - 1:** Flowchart of Arduino Codes	**35**
9.	**REFERENCES**	**39**

ABSTRACT

This internship report presents the design and development of two distinct yet complementary graphical user interfaces (GUIs): one for a pulsating frequency-based sensor system and another for a spirometer application. The pulsating sensor GUI was created to interface with frequency outputting sensors using an Arduino Mega 2560 and display real-time frequency data across four input channels. It incorporates various communication protocols such as UART, I^2C, and parallel LCD interfacing. Features such as live plotting (full and moving plots), statistical analysis, and Excel-based data saving were integrated using the PyQt5 framework.

The second part of the project focused on developing a GUI for a spirometer used in pulmonary diagnostics. The system employs a venturi tube and an ADP810 differential pressure sensor to measure flow rate and calculate key respiratory parameters like FEV1, FVC, and their ratio, crucial for diagnosing lung disorders like COPD and asthma. Real-time plots of Flow vs Time and Flow vs Volume were implemented, along with Excel-based data saving. This work showcases how embedded systems and GUI design can be synergistically used to develop user-friendly, medically significant applications.

List of tables

Table Name	Page Number
Table 1: Parameters in Spirometer Test	5
Table 2: TCCRxA Register	9
Table 3: TCCRxB Register	9
Table 4: Clock Select Bit Description	10
Table 5: TCCRxC Register	10
Table 6: TCNTx Register	10
Table 7: TIMSKx Register	11
Table 8: TIFRx Register	11
Table 9: Memory Mapping of ATMega 2560	11
Table 10: Control Register	14
Table 11: Frequency Select	14
Table 12: Channel 3 Log	27
Table 13: Channel 4 Log	29
Table 14: Statistics Log	32

List of figures

Figure Name	Page Number
Figure 1: 4 channel Frequency Meter block diagram	8
Figure 2: DS3231 Pinout	13
Figure 3: DS3231 Internal Block Diagram	13
Figure 4: TFT LCD Shield Pinout	14
Figure 5: Parts of Micro SD Card Module and its Pinout	15
Figure 6: 4 channel Frequency Meter Circuit Diagram	16
Figure 7: Schematic Diagram	16
Figure 8: Spirometer block diagram	17
Figure 9: Ideal Flow vs Volume graph	18
Figure 10: ADP810 Sensor Pinout	19
Figure 11: ADB810 Sensor block diagram	20
Figure 12: GUI functions for Frequency Meter	23
Figure 13: GUI functions for Spirometer	25
Figure 14: 4 channel Frequency Meter	26
Figure 15: GUI	26
Figure 16: GUI Statistics	29
Figure 17: Spirometer GUI	33

List of Symbols

Symbol	Meaning
f	Frequency
T	Time period of the signal
P_1	Pressure at inlet of venturi tube
P_2	Pressure at throat of venturi tube
v_1	Velocity of fluid at the inlet of venturi tube
v_2	Velocity of fluid at the throat of venturi tube
ρ	Density of the fluid
Q	Flow rate
A_1	Area of cross -section of inlet of venturi tube
A_2	Area of cross -section of throat of venturi tube
C	Capacitance
ε	Permittivity
A	Area of the plates
d	Distance between the plates

List of Abbreviations

Abbreviation	Full Form
Hz	Hertz (Unit of Frequency)
FVC	Forced Vital Capacity
FEV1	Forced Expiratory Volume in 1 sec
GUI	Graphical User Interface
PyQt5	Python Qt version 5
Qt	Qt framework (C++ framework)
IGCAR	Indira Gandhi Centre for Atomic Research
NTU	Nephelometric Turbidity Units
ATMega	Advanced Technology Mega (Microcontroller family)
RTC	Real Time Clock
LCD	Liquid Crystal Display
TFT	Thin Film Transistor
I^2C	Inter - Integrated Circuit
UART	Universal Asynchronous Receiver and Transmitter
CRC - 8	8 - bit Cyclic Redundancy Check
ASIC	Application - Specific Integrated Circuit
MEMS	Micro-Electro-Mechanical Systems
COPD	Chronic Obstructive Pulmonary Disease

1. OVERVIEW OF IGCAR

1.1 IGCAR [1]

It is the second largest establishment of the Department of Atomic Energy next to Bhabha Atomic Research Centre, was set up at Kalpakkam, 80 KMs south of Chennai (MADRAS), in 1971 with the main objective of conducting broad based multidisciplinary programme of scientific research and advanced Engineering, directed towards the development of sodium cooled Fast Breeder Reactor [FBR] technology, in India. This is part of the second stage of Indian Atomic Energy Programme, which is aimed at preparing the country for utilization of the extensive Thorium reserves and providing means to meet the large demands of electrical energy in the 21st century. In meeting the objectives, a modest beginning was made by constructing a sodium cooled Fast Breeder Test Reactor [FBTR], with a nominal power of 40 MW, based on the French Reactor, RAPSODIE. The reactor attained its first criticality on 18th Oct, 1985 and has been in operation at its maximum attainable power level of 10.5 MW with a small core. It is the first of its kind in the world to use Plutonium Uranium mixed carbide as a driver fuel. Over the years, the centre has established comprehensive R & D facilities covering the entire spectrum of FBR technology related to Sodium Technology, Reactor Engineering, Reactor Physics, Metallurgy and Materials, Chemistry of Fuels and its materials, Fuel Reprocessing, Reactor Safety, Control and Instrumentation, Computer Applications etc., and has developed a strong base in a variety of disciplines related to this advanced technology. With the experience and expertise gained by the successful operation of FBTR, the Centre has embarked upon the design and construction of 500 MWe, Prototype Fast Breeder Reactor [PFBR]. Various R & D activities in the areas of Structural Mechanics, Thermal Hydraulics and flow induced vibration, Component Testing in high temperature sodium environment, sodium-water reaction, hydraulic development of sodium pumps etc., were pursued and the design was completed. The PFBR is under advanced stage of construction and commissioning by BHAVINI.

As a part of efforts for closing the fuel cycle, a Fast Reactor Fuel Reprocessing Plant is under construction.

A 30 KW, U233 fueled mini reactor, KAMINI has been made operational for neutron radiography, neutron activation analysis etc., IGCAR utilizes its expertise and resources in enhancing its standing as a leading Centre of research in various branches of basic, applied and engineering sciences that have a bearing on Nuclear Technology like Structural Mechanics, Heat and Mass Transfer, Material Science, Fabrication Processes, Non-Destructive Testing, Chemical sensors, High temperature thermodynamics, Radiation Physics, Computer science etc.,

Apart from thrust areas related to nuclear technology, the Centre has credentials as a leader of research in various frontier and topical subjects like Quasicrystals, Oxide superconductors, Nano-structures, clusters, SQUID fabrication programs, exopolymers and experimental simulation of condensed matter using colloids etc., IGCAR has extended its expertise and facilities to other vital sectors as Defence, Space and other industries of India to develop techniques for reliable solutions to specialized problems. It has collaborations with educational and R & D institutes like Indian Institutes of Technology, Indian Institute of Science, Pilani, Regional Engineering Colleges, National Research Laboratories, Public Units and Institutes abroad.

A modern Library comprising 62,000 volumes of books, 41,000 back volumes, about 920 journals and 1.95 lakhs reports in all disciplines caters to the technical needs of the Scientists and Engineers. The Central Workshop is fully equipped with sophisticated machines for the fabrication of precision components.

The Computer Division houses Silicon Graphics Power Challenge L servers, SGI workstations, 8 Noded Xeon Servers to meet the computational demands of the users. The centre has sanctioned staff strength of 2814 including 1187 Engineers and Scientists. The annual outlay of the Centre is around 754.47 crore rupees towards its activities and plan.

Shri C G Karhadkar, Distinguished Scientist had taken over as Director, IGCAR on 01.06.2024.

1.2 Publications by Dr. Ramesh Sanga ISS/SISD/EIG

- Analysis and Design of a Self-Oscillating Opto Resistive Type Quasi Digital Sensor to Check the Quality of Lubricant Oil
- Design of an Analog Wireless Communication for Radiation Monitoring System with Inherent Encryption
- Design and Development of a Capacitive Type Quasi Digital Sensor and Instrument to In-situ monitoring of viscosity of Lubricant Oil
- Design and Development of Quasi Digital sensor-based Spirometer
- Deployment of the inductance-based Quasi Digital Sensor and Instrument to Monitor the Metallic Liquid Level

2. INTRODUCTION

IGCAR is responsible for design of fast breeder reactors. In this, the Electronics and Instrumentation Group is responsible for design and development of sensors and instrumentation, embedded systems for successful operation of the fast breeder reactor.

In the Innovative Sensors Section, SISD\EIG, various types of sensors are being developed and deployed for many applications in and around Kalpakkam and other DAE facilities. These sensors are designed by utilizing electrical parameters like resistance, capacitance, inductance and electric potential.

The main applications are measurement of level, temperature, conductivity, pH, water turbidity etc.

These sensors output is in the form of TTL pulse in response to a tiny change in the parameter to be measured. By measuring the pulse frequency and correlating with the sensing parameter is necessary to measure the process parameter and give annunciation to the plant operator to operate the plant effectively.

Since the output is in the form of TTL pulses there is a requirement for development of embedded system, firmware and GUI to read the sensor output, convert into the process parameter being measured and display it.

Currently, in this section, embedded systems and instruments are required for measuring signal from sensors and converting it to process parameters and displaying to the operator using Keil software, Orcad, microC, etc. These softwares are continually upgraded to newer versions and hence need upgraded versions of operating systems. These softwares are to be procured from different companies and one has to depend on that. Currently, open-source tools (like Arduino, Python, VSCode, etc) are used instead of these. Design and development of instruments using these open-source tools is cost effective and availability of multiple vendors eliminates the obsolescence of components.

2.1 Objective

This project report aims to provide the reader with understanding of the nuances of designing and implementing an embedded system and how we can interface a graphical interface with it. We use an Arduino Mega powered by ATMega2560 processor to calculate the frequency of the 4 inputted waves and python based GUI using PyQt5 for developing a user interface. And also a spirometer using a venturi tube, ADP810 pressure sensor and an Arduino UNO powered by ATMeaga328P processor and its own GUI in python.

2.2 Spirometer

A spirometer is a device used to measure how well your lungs are working, how much air you inhale and exhale and how quickly you can do it. The parameters measured through the spirometer are

Table 1: Parameters in Spirometer Test

Parameter	**Description**
Forced Vital Capacity (FVC)	Total volume of air you can exhale forcefully after a deep breath
Forced Expiratory Volume in 1 sec (FEV1)	Amount of air exhaled in the first second of the FVC test
FEV1/FVC ratio	Helps distinguish between obstructive and restrictive lung conditions
Breathing Rate	No. of breaths per minute
Flow Rates	Speed at which air is moved in and out

In short, a spirometer is like a thermometer for your lungs — it shows how strong, healthy, or impaired your breathing is.

2.3 GUI and PyQt5 Overview [2]

A PyQt5 framework along with python coding is used to create the required UIs. PyQt5 is a set of Python bindings for Qt5, a C++ framework. It allows you to create graphical user interfaces (GUIs) in Python.

The commonly used GUI layout are:

QVBoxLayout - vertical layout
QHBoxLayout - horizontal layout
QGridLayout - grid-based layout
QFormLayout - label-input pair layout

And commonly used widgets are:

QLabel - display text or images
QPushButton - button
QLineEdit - text input field
QTextEdit - mulit-line text input
QComboBox - dropdown menu
QCheckBox - checkbox
QRadioButton - radiobutton
QTableWidget - table/grid

2.4 Communication Protocols Used

1. **Between Arduino and TFT LCD Shield**

   The type of communication used is 8080-style 8-bit parallel mode of communication. It has 8 data lines D0-D7 and 5 control lines (WR, RD, RS, CS, RESET).

   RD is often held high and the CS is active low.

   To read a command, RS = 0, and put the command in D0-D7 and pull WR to low.

   To read a data, RS = 1, and put the data byte in D0-D7 and pull WR to low.

2. **Between Arduino and DS3231**

   It uses I^2C communication, which uses two lines, SCL (clock) and SDA (data). You have a master (Arduino) and one or more slaves (DS3232).
   Master initiates communication. Each slave has a 7-bit address
   (DS3231 = 0x68). Data is exchanged using the Wire library with start/stop conditions.

3. **Between Arduino and PC**

   This uses UART (universal asynchronous receiver and transmitter). It is a two-wire asynchronous protocol (TX for transmission and RX for receiving). Arduino Mega uses TX1 and RX1 for UART. Since it is asynchronous, there is no clock line, meaning that the devices must have the same baud rate.

3. DESIGN AND DEVELOPMENT OF EMBEDDED SYSTEMS FOR PULSATING SENSOR

The monitoring of water quality parameters such as pH, turbidity and temperature, is critical in the reservoir of the power plant at Kalpakkam. This chapter focuses on the design and development of an embedded system that integrates sensors and processing units for real-time measurement and monitoring of these parameters. The sensor at ISS gives the output in the form of TTL pulses. The work focuses on designing a four-channel instrument for reading the TTL pulse frequencies from four sensors.

3.1 Embedded Hardware Design

The hardware consists of Arduino Mega 2560 microcontroller, RTC module, SD card module, LCD module, and UART communication for designing four-channel instrument. Figure.1 shows the block diagram of the embedded system. The heart of the instrument is the development of frequency counters. The timers in the microcontroller are used for frequency counting.

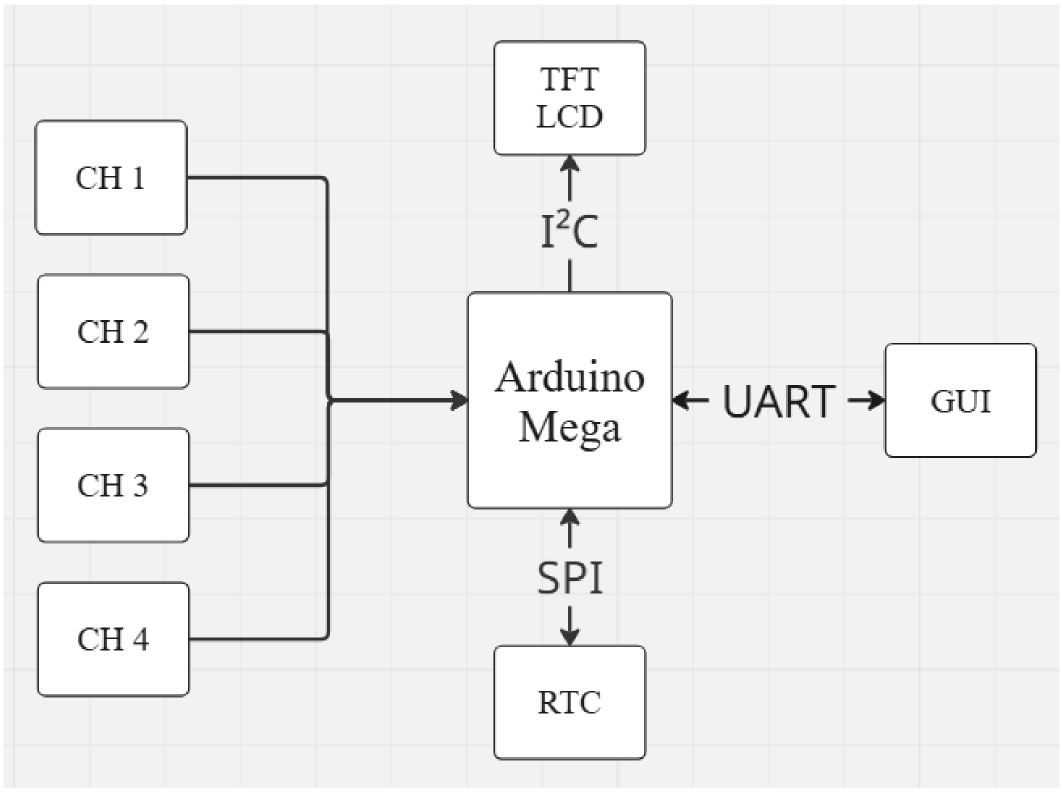


Figure 1: 4 channel Frequency Meter block diagram

3.2 Timers in Arduino Mega 2560 [3]

Arduino Mega 2560 is a microcontroller with an internal clock of 16MHz. It has two 8-bit timers T0 and T2 and four 16-bit timers T1, T3, T4 and T5.

T0 is used by the Arduino to execute functions such as millis(), micros(), delay(), delayMicroseconds(). So, using timer 0 will affect the operation of these functions.

T2 is used by the Arduino to execute the tone() function. So, it is also risky to use. Moreover, it is an 8-bit timer, which has lesser resolution compared to the 16- bit timers.

So we have used the timers T1, T3, T4 and T5 for inputs.

The 16- bit timers are controlled using the TCCRxA, TCCRxB, TCCRxC, TCNTx, OCRxA, OCRxB, OCRxC, TIMSKx, TIFRx where x is the timer number.

The three channels A, B and C are used in a single timer, so that a timer can perform different output functions at the same time. Like channel A can be used to generate a 25% duty cycle, B can be used to generate 50% duty cycle and C can for generating 75% duty cycle using the same timer in output compare mode.

TCCRxA (Timer/Counter Control Register A)

Table 2: TCCRxA Register

7	6	5	4	3	2	1	0
COMxA1	COMxA0	COMxB1	COMxB0	COMxC1	COMxC0	WGMx1	WGMx2

Bit 7:6 – COMxA1:0: Compare Output Mode for Channel A

Bit 5:4 – COMxB1:0: Compare Output Mode for Channel B

Bit 3:2 – COMxC1:0: Compare Output Mode for Channel C

Bit 1:0 – WGMx1:0: Waveform Generation Mode

TCCRxB (Timer/Counter Control Register B)

Table 3: TCCRxB Register

7	6	5	4	3	2	1	0
ICNCx	ICESx	-	WGMx3	WGMx2	CSx2	CSx1	CSx0

Bit 7 – ICNCx: Input Capture Noise Canceler

Bit 6 – ICESx: Input Capture Edge Select

Bit 5 – Reserved Bit

Bit 4:3 – WGMx3:2: Waveform Generation Mode

Bit 2:0 – CSx2:0: Clock Select

Table 4: Clock Select Bit Description

CSx2	CSx1	CSx0	Description
0	0	0	No clock source. (Timer/Counter stopped
0	0	1	clkI/O/1 (No prescaling
0	1	0	clkI/O/8 (From prescaler)
0	1	1	clkI/O/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	clkI/O/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

TCCRxC (Timer/Counter x Control Register C)

Table 5: TCCRxC Register

7	6	5	4	3	2	1	0
FOCxA	FOCxB	FOCxC	-	-	-	-	-

Bit 7 – FOCxA: Force Output Compare for Channel A

Bit 6 – FOCxB: Force Output Compare for Channel B

Bit 5 – FOCxC: Force Output Compare for Channel C

Bit 4:0 – Reserved Bits

TCNTx (Timer/Counter x)

Table 6: TCNTx Register

7	6	5	4	3	2	1	0
TCNTx[15:8]							
TCNTx[7:0]							

The TNCT has the running timer value. Since the timers are 16-bit, you have TCNTxH and TCNTxL.

TIMSKx

Table 7: TIMSKx Register

7	6	5	4	3	2	1	0
-	-	ICIEx	-	OCIExC	OCIExB	OCIExA	TOIEx

Bit 5 – ICIEx: Timer/Counter x, Input Capture Interrupt Enable

Bit 3 – OCIExC: Timer/Counter x, Output Compare C Match Interrupt Enable

Bit 2 – OCIExB: Timer/Counter x, Output Compare B Match Interrupt Enable

Bit 1 – OCIExA: Timer/Counter x, Output Compare A Match Interrupt Enable

Bit 0 – TOIEx: Timer/Counter x, Overflow Interrupt Enable

TIFRx

Table 8: TIFRx Register

7	6	5	4	3	2	1	0
-	-	ICF1	-	OCFxC	OCFxB	OCFxA	TOVx

Bit 5 – ICFx: Timer/Counter x, Input Capture Flag

Bit 3– OCFxC: Timer/Counter x, Output Compare C Match Flag

Bit 2 – OCFxB: Timer/Counter x, Output Compare B Match Flag

Bit 1 – OCFxA: Timer/Counter x, Output Compare A Match Flag

Bit 0 – TOVx: Timer/Counter x, Overflow Flag

Table 9: Memory Mapping of ATMega 2560

Hex address	Register name
0x125	TCNT5[15:8]
0x124	TCNT5[7:0]
0x123	Reserved
0x122	TCCR5C
0x121	TCCR5B
0x120	TCCR5A
0xA5	TCNT4[15:8]

0xA4	TCNT4[7:0]
0xA3	Reserved
0xA2	TCCR4C
0xA1	TCCR4B
0xA0	TCCR4A
0x95	TCNT3[15:8]
0x94	TCNT3[7:0]
0x93	Reserved
0x92	TCCR3C
0x91	TCCR3B
0x90	TCCR3A
0x85	TCNT1[15:8]
0x84	TCNT1[7:0]
0x83	Reserved
0x82	TCCR1C
0x81	TCCR1B
0x80	TCCR1A
0x73	TIMSK5
0x72	TIMSK4
0x71	TIMSK3
0x70	
0x6F	TIMSK1
0x3A	TIFR5
0x39	TIFR4
0x38	TIFR3
0x37	
0x36	TIFR1

Here, we are using Timer 1, 3, 4 and 5 as input. Hence the input pins (according to ATMega2560 datasheet) are PD6, PE6, PH7 and PL2. But according to the arduino mega 2560 schematic, only the timer 5 pin is connected to D47. The rest are not connected. So to use the other 3 channels, you will have to solder the pins from the ATMega 2560 processor.

<u>**3.3 DS3231 RTC**</u> [4]

The RTC has an internal TCXO (Temperature Controlled Crystal Oscillator) of 32kHz, which is directly fed to the 32kHz output pin. To get an output of 1Hz or 1kHz or 4kHz or 8kHz, the clock is divided using a divider circuit, based on the control register.

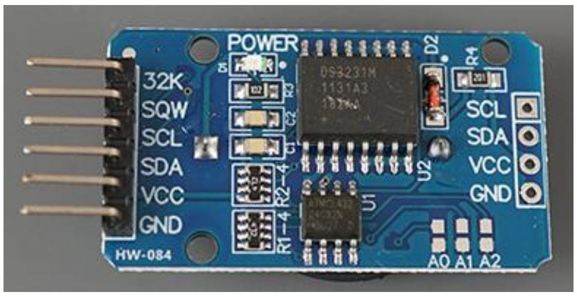


Figure 2: DS3231 Pinout

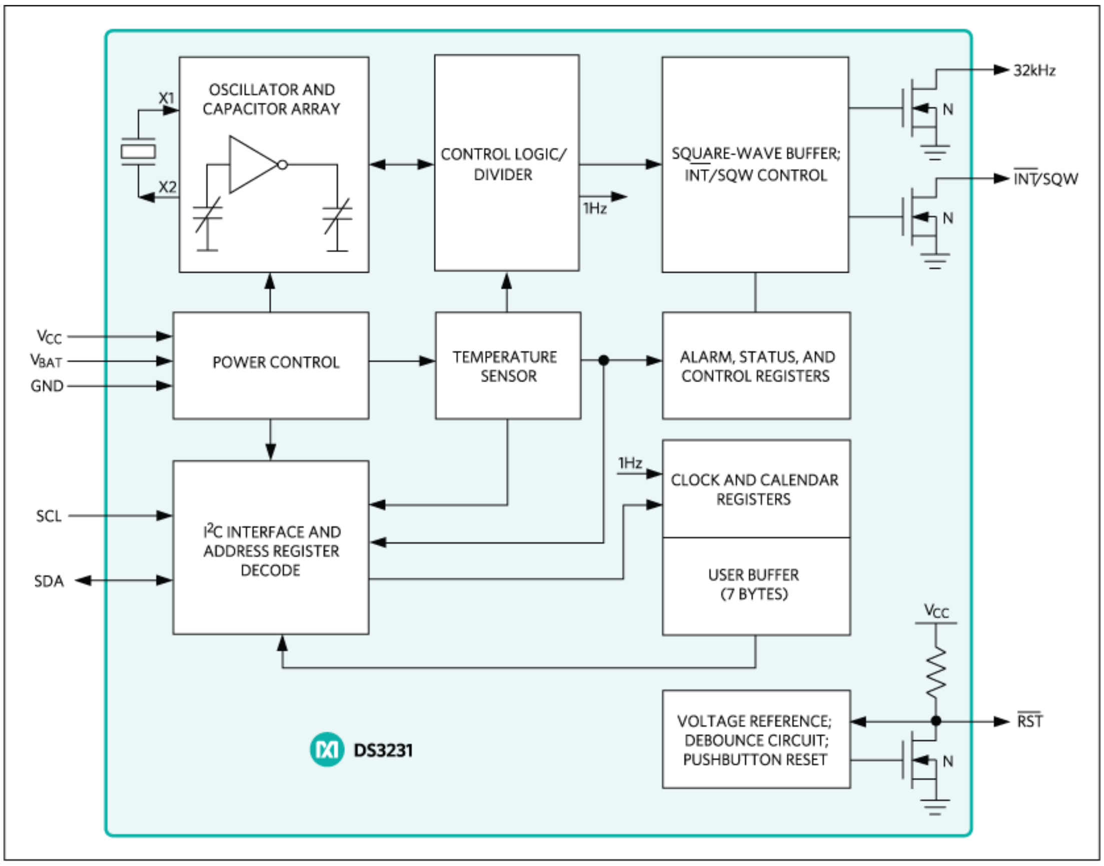

X1
X2
OSCILLATOR AND CAPACITOR ARRAY
CONTROL LOGIC/ DIVIDER
1Hz
SQUARE-WAVE BUFFER; INT/SQW CONTROL
32kHz
N
INT/SQW
N
VCC
VBAT
GND
POWER CONTROL
TEMPERATURE SENSOR
ALARM, STATUS, AND CONTROL REGISTERS
1Hz
CLOCK AND CALENDAR REGISTERS
SCL
SDA
I²C INTERFACE AND ADDRESS REGISTER DECODE
USER BUFFER (7 BYTES)
VCC
VOLTAGE REFERENCE; DEBOUNCE CIRCUIT; PUSHBUTTON RESET
RST
N
DS3231

Figure 3: DS3231 Internal Block Diagram

Control Register (0x0E)

Table 10: Control Register

7	6	5	4	3	2	1	0
$\underline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE

Bit 7: Enable Oscillator (EOSC)

Bit 6: Battery-Backed Square-Wave Enable (BBSQW)

Bit 5: Convert Temperature (CONV)

Bits 4 and 3: Rate Select (RS2 and RS1)

Bit 2: Interrupt Control (INTCN)

Bit 1: Alarm 2 Interrupt Enable (A2IE)

Bit 0: Alarm 1 Interrupt Enable (A1IE)

Table 11: Frequency Select

RS2	RS1	Square Wave Output Frequency
0	0	1Hz
0	1	1.024kHz
1	0	4.096kHz
1	1	8.192kHz

<u>3.4 TFT LCD Shield</u> [5]

The LCD used is a 2.4" TFT LCD SHIELD with the ST7789 driver IC. The MCUFRIEND_kbv header file is used to interface in 8-bit mode parallely with the Arduino, even though the driver IC supports only SPI. The Adafruit_GFX library is used for the functions like fillScreen(), setCursor(), etc.

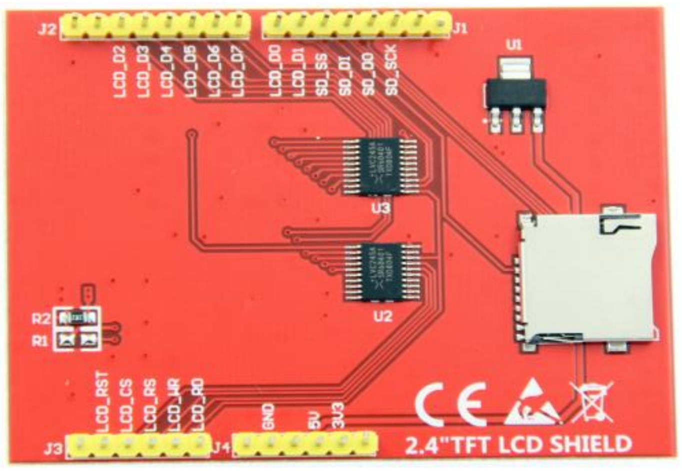


Figure 4: TFT LCD Shield Pinout

<u>3.5 Micro SD Card Module</u> [6]

Attach the SD Card to the module and interface it with the Arduino Mega (Refer Figure 5) Run the Arduino example code. If the **“”** line is printing in the serial monitor, recheck the wiring. If the **“”** line is printing, format the SD Card and try again. If the **“”** line is printing, then use can proceed with further coding.

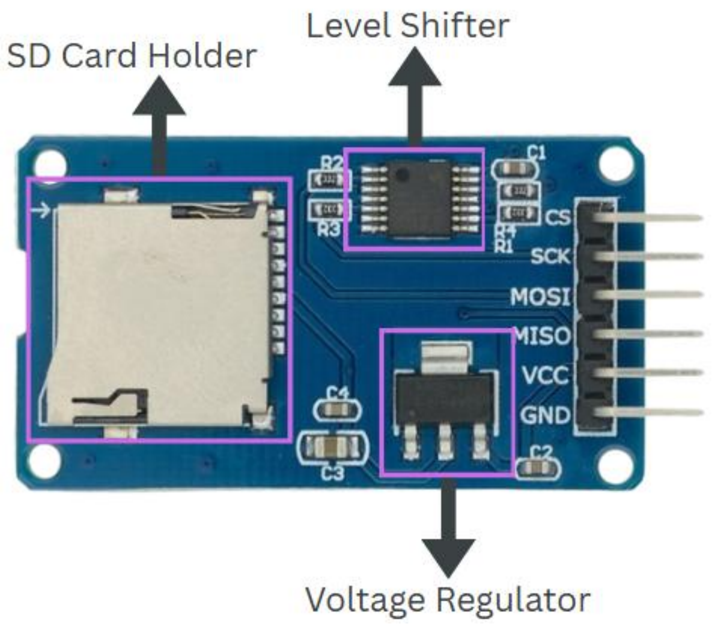


Figure 5: Parts of Micro SD Card Module and its Pinout

There are only three components that are significant, first is the Micro SD Card Holder Itself. This holder makes it easy for us to swap between different SD card modules. The second most important thing is the level shifter IC as the SD card module runs only on 3.3V and it has a maximum operating voltage of 3.6V so if we directly connect the SD card to 5V it will definitely kill the SD card. Also, the module has an onboard ultra-low dropout regulator that will convert the voltage from 5V to 3.3V. That is also why this module can operate on 3.3V power

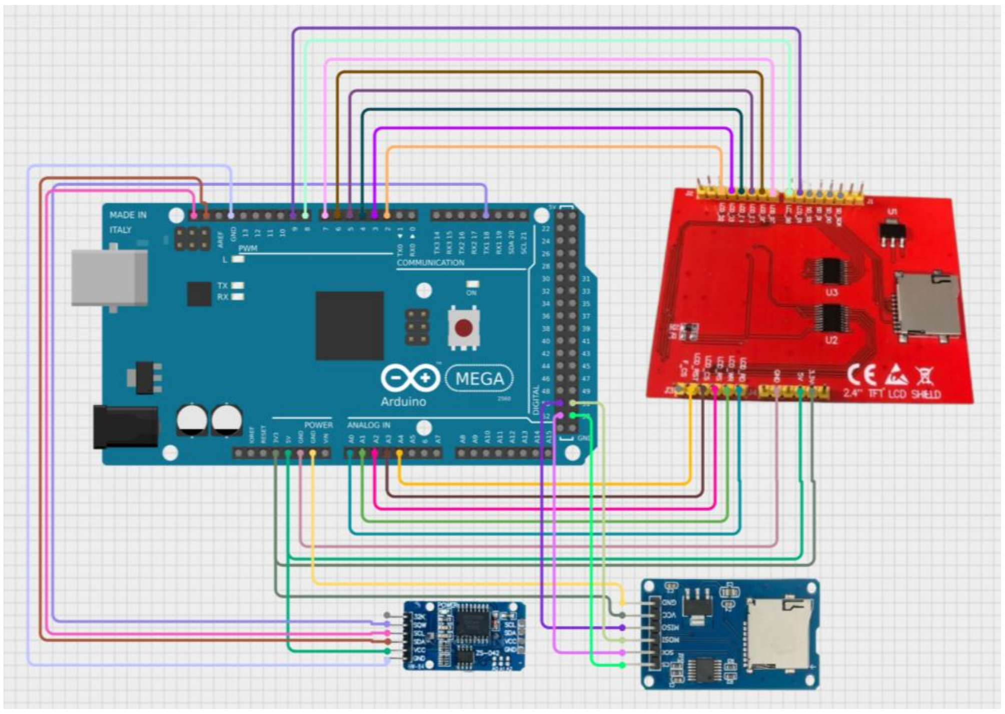


Figure 6: 4 channel Frequency Meter Circuit Diagram

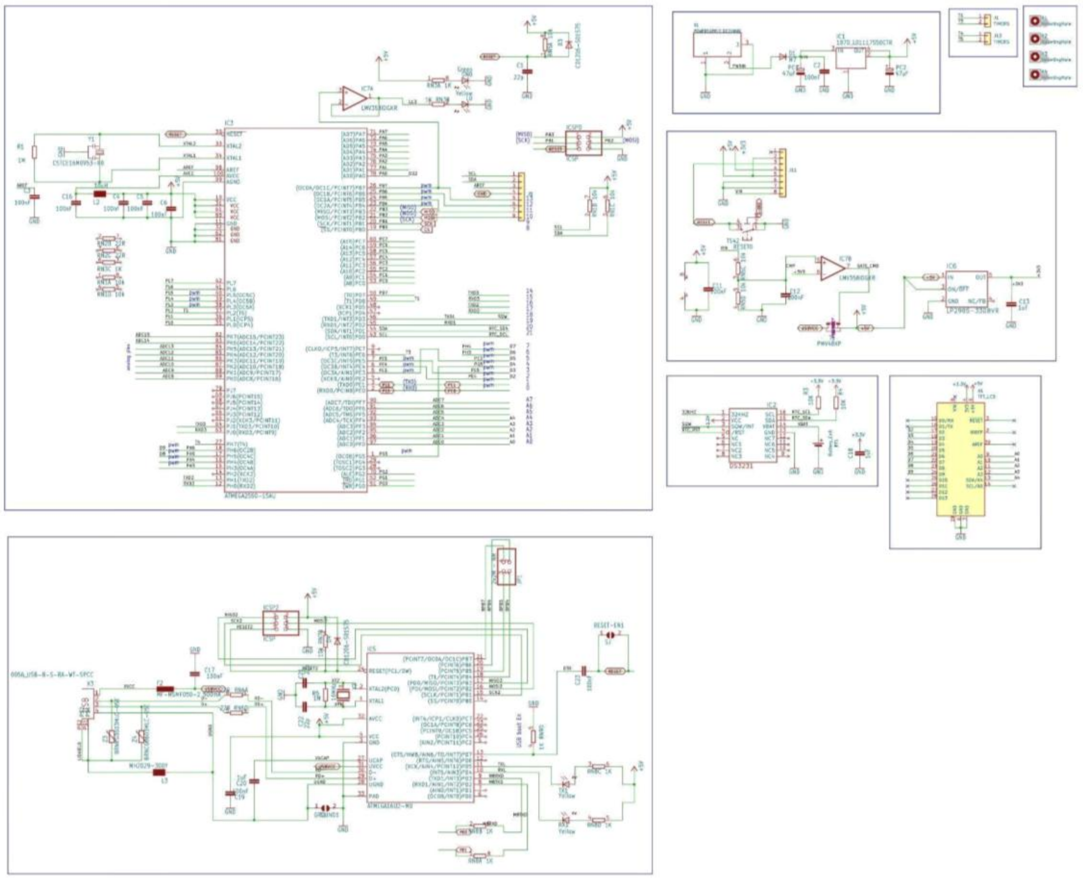


Figure 7: Schematic Diagram [7]

4. DESIGN AND DEVELOPMENT OF EMBEDDED SYSTEMS FOR SPIROMETER

This chapter focuses on a specific application, the spirometer. When a person blows through the venturi tube, the pressure difference between the inlet and throat is measured by the pressure sensor which is received by the Arduino UNO and using UART communicates with the GUI, which does the further processing of data and displays the required parameters. These parameters, namely, FVC, FEV1 and FEV1/FVC ratio helps the medical personnel identify diseases in the lungs.

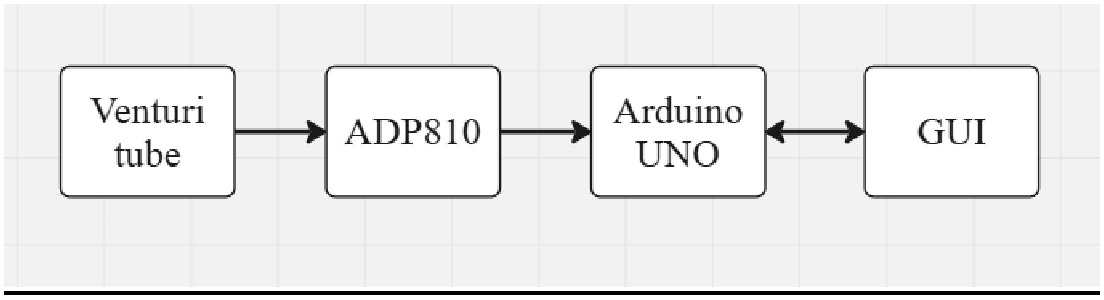


Figure 8: Spirometer block diagram

The FVC (Forced Vital Capacity) is the volume that can be forcefully blown out after full inspiration, measured in liters.

The FEV1 (Forced Expiratory Volume 1) is the volume of air that can forcibly be blown out in the first 1-second, after full inspiration.

The FEV1: FVC ratio in adults should be 70% - 80% and in young adults and children, >80%.

In obstructive diseases (asthma, COPD, chronic bronchitis, emphysema) FEV1 is diminished because of increased airway resistance to expiratory flow; the FVC may be decreased as well, due to the premature closure of airway in expiration, just not in the same proportion as FEV1 (for instance, both FEV1 and FVC are reduced, but the former is more affected because of the increased airway resistance). This generates a reduced value (<70%, often ~45%).

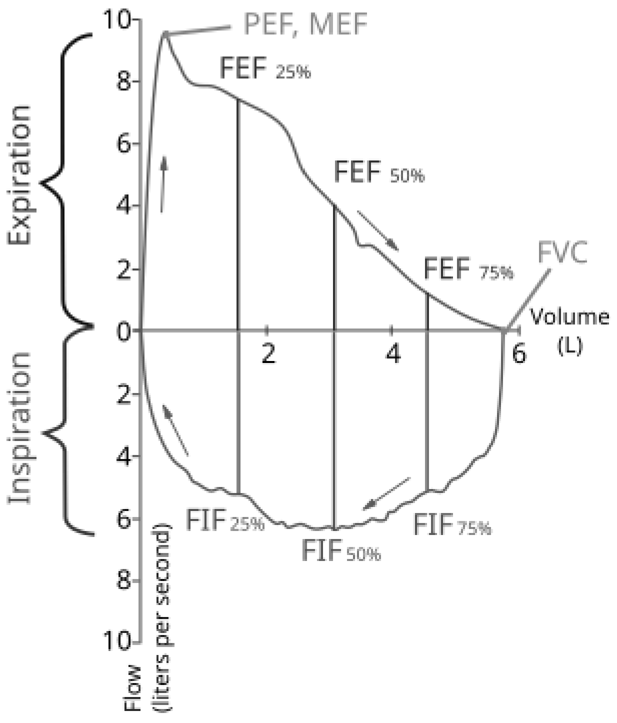


Figure 9: Ideal Flow vs Volume graph [8]

4.1 Venturi tube

The venturi tube is a tube that diverges at two ends and converges in the centre.
According to the Bernoulli's principle,

$$P_1 + \frac{1}{2}\rho {v_1}^2 = P_2 + \frac{1}{2}\rho {v_2}^2$$

When fluid enters the narrow throat, it speeds up. Since energy is conserved, the velocity increases and pressure drops. This pressure drop is proportional to the flow rate.

$$Q = A_2 * v_2 = A_2 * \sqrt{\frac{2\,|P_1 - P_2|}{\rho\,(1 - \frac{A_2}{A_1})^2}}$$

$$Q = C * A_1 * \sqrt{\frac{|P_1 - P_2|}{\rho}}$$

where $$C = \frac{A_2}{A_1}\sqrt{\frac{2}{(1-\frac{A_2}{A_1})^2}}$$

4.2 ADP810 Sensor

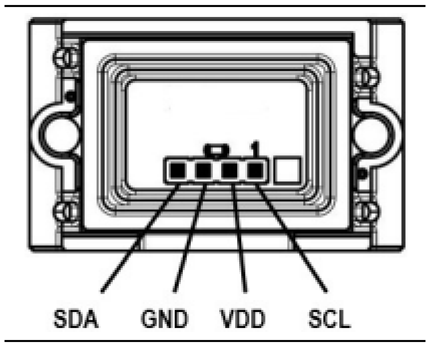


Figure 10: ADP810 Sensor Pinout

Working

The sensor has two tubes, through which input should be given. At the core is a microscopic flexible diaphragm (usually made of silicon). This diaphragm separates the two pressure chambers. The MEMS diaphragm bends in accordance with the difference in pressure between the chambers. And the diaphragm is a part of a capacitor, whose one plate is a fixed electrode and other a flexible diaphragm. When the pressure changes, the diaphragm bends, changing the distance between the two plates of capacitor and hence the capacitance value changes.

$$C = \varepsilon\frac{A}{d}$$

This change in capacitance is very small (femtofarads) but measurable. The sensor includes an on-chip ASIC which converts this analog capacitance change into a digital signal, applies calibration, temperature compensation, linearization and sends the result as digital output via I^2C. The analog capacitance value is mapped to a pressure value using a calibration curve.

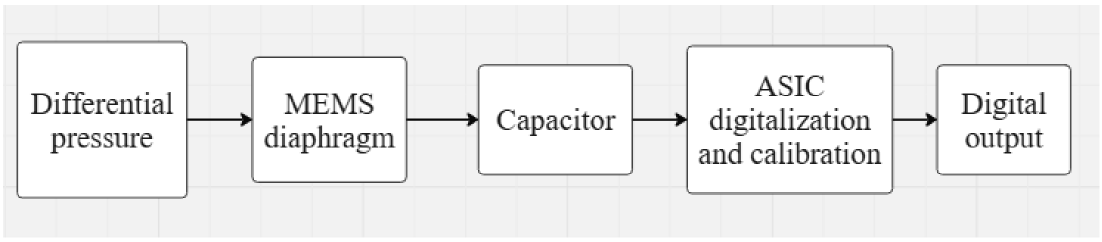


Figure 11: ADB810 Sensor block diagram

CRC - 8 algorithm

CRC is the Cyclic Redundancy Check used to check the integrity of received data. This data is received after two bytes of input data. The CRC - 8 algorithm is used to verify the integrity. CRC-8 is an algorithm that treats the data as a binary stream, divides it by a polynomial using XOR (not arithmetic division). The remainder after division is the CRC. If both the CRC values match, there is no corruption in the data.

According to the ADP810 datasheet, the initial value of CRC is 0xFF and the polynomial is 0x31.

4.3 Arduino UNO and Methodology

The ADP810 sensor is interfaced with an Arduino UNO and the microcontroller receives the pressure and temperature data from the sensor. It verifies its integrity using the CRC - 8 algorithm and sends the data to GUI through the serial monitor.

5. DESIGN AND DEVELOPMENT OF GUI FOR EMBEDDED SYSTEMS

5.1 GUI for pulsating sensor

Port settings

This group box contains the settings related to the UART communication between the PC and Arduino. You have to select the COM ports from the available ports shown in a dropbox and choose the baud rate, also given in a dropbox.

Note: Your selected baud rate must be the same that mentioned in the arduino code, Serial.begin()

Graph settings

When a frequency outputting sensor is interfaced with the arduino, the arduino measures its frequency and sends it to GUI The GUI has to find the required parameter from the frequency value. So you have to choose whether you want to plot the Frequency vs Time plot or Parameter vs Time plot.

Note: By default, the Frequency vs Time plot will be selected.

Probe parameters

To calculate Parameter = $Af^3 + Bf^2 + Cf + D$, where f is the frequency, you have to enter the coefficients A, B, C and D, which will be entered in the probe parameter block.

Note: You have to enter the coefficient value for each channel separately. By default, all coefficient values will be 0.

Time Least Count

This is a text input box where you have to enter the sampling time (the time interval at which you would like the arduino to process the frequency) and plot the data in the GUI. You have to also choose whether the time entered is in milliseconds or seconds.

Note: By default, the seconds radio button will be choosen.

Full plot and moving plot

Since we have 4 different channels, I have designed a full plot which will be a full time plot plotting the frequency values of all the 4 channels. And 4 individual plots, one for each channel. The individual plots will be moving, that is, it will plot the last few values only, so that the recent pulses can be viewed clearly.

Note: If you have chosen the parameter plot, the full plot will plot the frequency values while the individual plot will plot the parameter values.

Max points

In moving plots, the last few values will be plotted. The "last few" value will be decided by "max points", which the user will have to enter.

Note: By default the max points value will be 10, that is the last 10 points will be plotted in every moving plot.

Start plotting

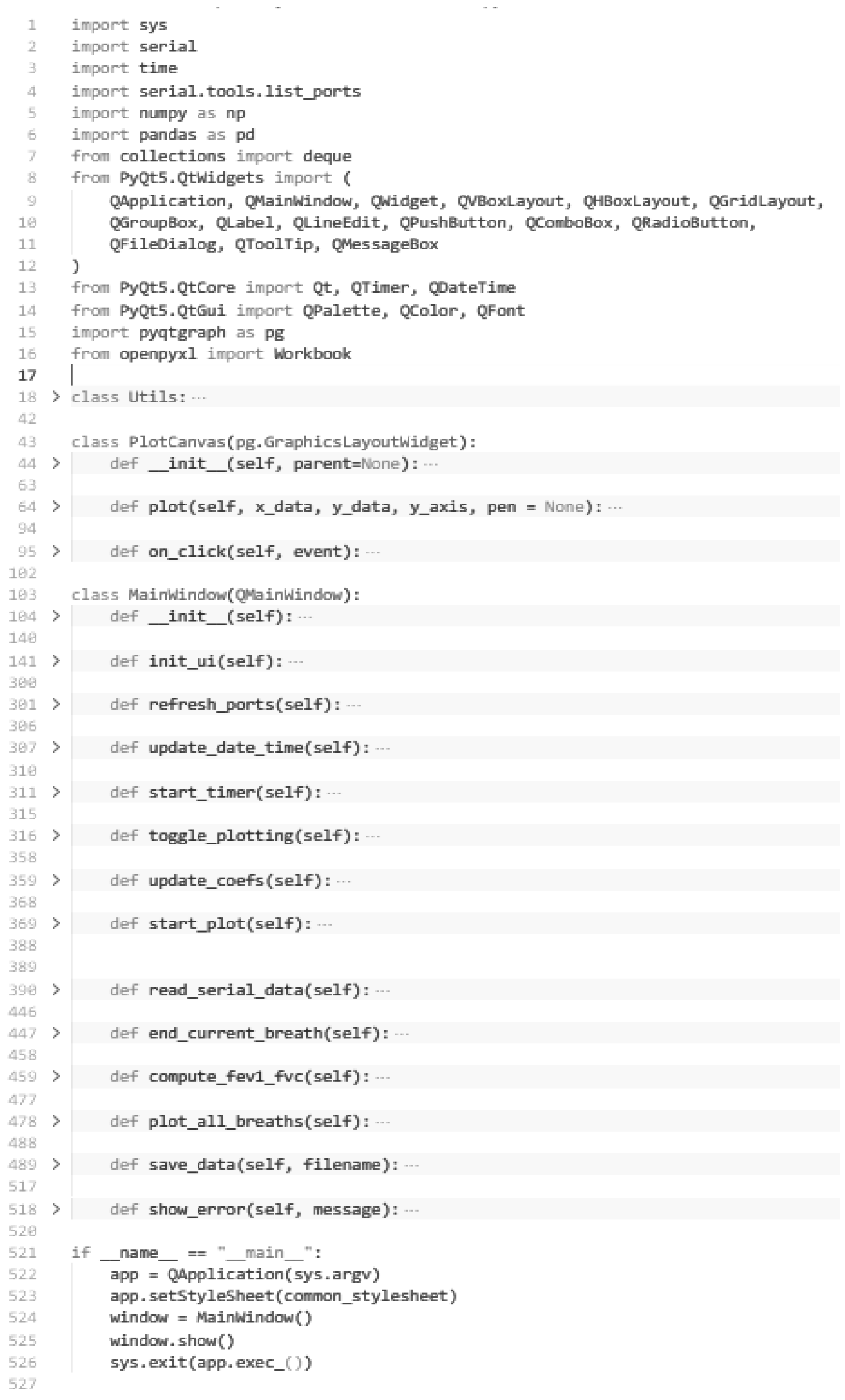
On clicking the start plotting button, the GUI sends the Time Least Count value to the Arduino to start processing the inputs. And the button text will change to "STOP PLOTTING" and on clicking it, all the plots stop and the serial communication port is closed. And the button reverts back to "START PLOTTING" and on clicking on it again, you are restarting the process, so the entire plots start from 0 again. You can also change the graph settings, probe parameters and time least count before clicking on "START PLOTTING".

Show stats

On clicking this button, irrespective of whether the GUI is plotting or not, it will display the statistical parameters of the frequency and parameter of each channel. And to calculate the stats, it will use the data available until you press "Show stats"; it won't be able to keep on updating the stats as each data comes in. It will process it only when you click the button. The statistical parameters displayed will be its mean, minimum, maximum, standard deviation, skewness and kurtosis.

Save data

On clicking this button, a save file dialog box will open. Choose the destination and the file name you want to store the data in. As we have 4 different channels, you will have different sheets to store data for each channel and an additional sheet to store the statistical parameters. The statistical parameters will be calculated when you click on the “Save data” button. An additional feature is that the data stored will be from the very first click of “START PLOTTING”, that is, the previous plot data will also be stored. But the statistical parameters will be of the latest “START PLOTTING” values. Note: The data will be stored in excel document format.



```
import sys
import serial
import time
import serial.tools.list_ports
import numpy as np
import pandas as pd
from collections import deque
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QGridLayout,
    QGroupBox, QLabel, QLineEdit, QPushButton, QComboBox, QRadioButton,
    QFileDialog, QToolTip, QMessageBox
)
from PyQt5.QtCore import Qt, QTimer, QDateTime
from PyQt5.QtGui import QPalette, QColor, QFont
import pyqtgraph as pg
from openpyxl import Workbook

class Utils: ···

class PlotCanvas(pg.GraphicsLayoutWidget):
    def __init__(self, parent=None): ···

    def plot(self, x_data, y_data, y_axis, pen = None): ···

    def on_click(self, event): ···

class MainWindow(QMainWindow):
    def __init__(self): ···

    def init_ui(self): ···

    def refresh_ports(self): ···

    def update_date_time(self): ···

    def start_timer(self): ···

    def toggle_plotting(self): ···

    def update_coefs(self): ···

    def start_plot(self): ···


    def read_serial_data(self): ···

    def end_current_breath(self): ···

    def compute_fev1_fvc(self): ···

    def plot_all_breaths(self): ···

    def save_data(self, filename): ···

    def show_error(self, message): ···

if __name__ == "__main__":
    app = QApplication(sys.argv)
    app.setStyleSheet(common_stylesheet)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

Figure 12: GUI functions for Frequency Meter

5.2 GUI for spirometer

Port settings

This group box contains the settings related to the UART communication between the PC and Arduino. You have to select the COM ports from the available ports shown in a dropbox and choose the baud rate, also given in a dropbox.

Note: Your selected baud rate must be the same that mentioned in the arduino code, Serial.begin()

Parameters

To calculate the flow rate from pressure, we need the value of constant C, area of cross-section (cm^2) of outlet and density of the fluid (kg/m^3). So you have defined text input boxes to enter the specifications of your spirometer.

Start plotting

On clicking the start plotting button, the GUI starts processing the input received through UART. And the button text will change to “STOP PLOTTING” and on clicking it, all the plots stop and the serial communication port is closed. And the button reverts back to “START PLOTTING” and on clicking on it again, you are restarting the process, so the entire plots start from 0 again. You can also change the settings before clicking on “START PLOTTING”.

Flow vs Time plot

This plot plots the flow vs time graph. As you inhale and exhale, the pressure drops and then increases to a peak value. And on the next inhalation, the pressure drops again. Since flow is directly proportional to the pressure difference, it gives a square wave-like plot..

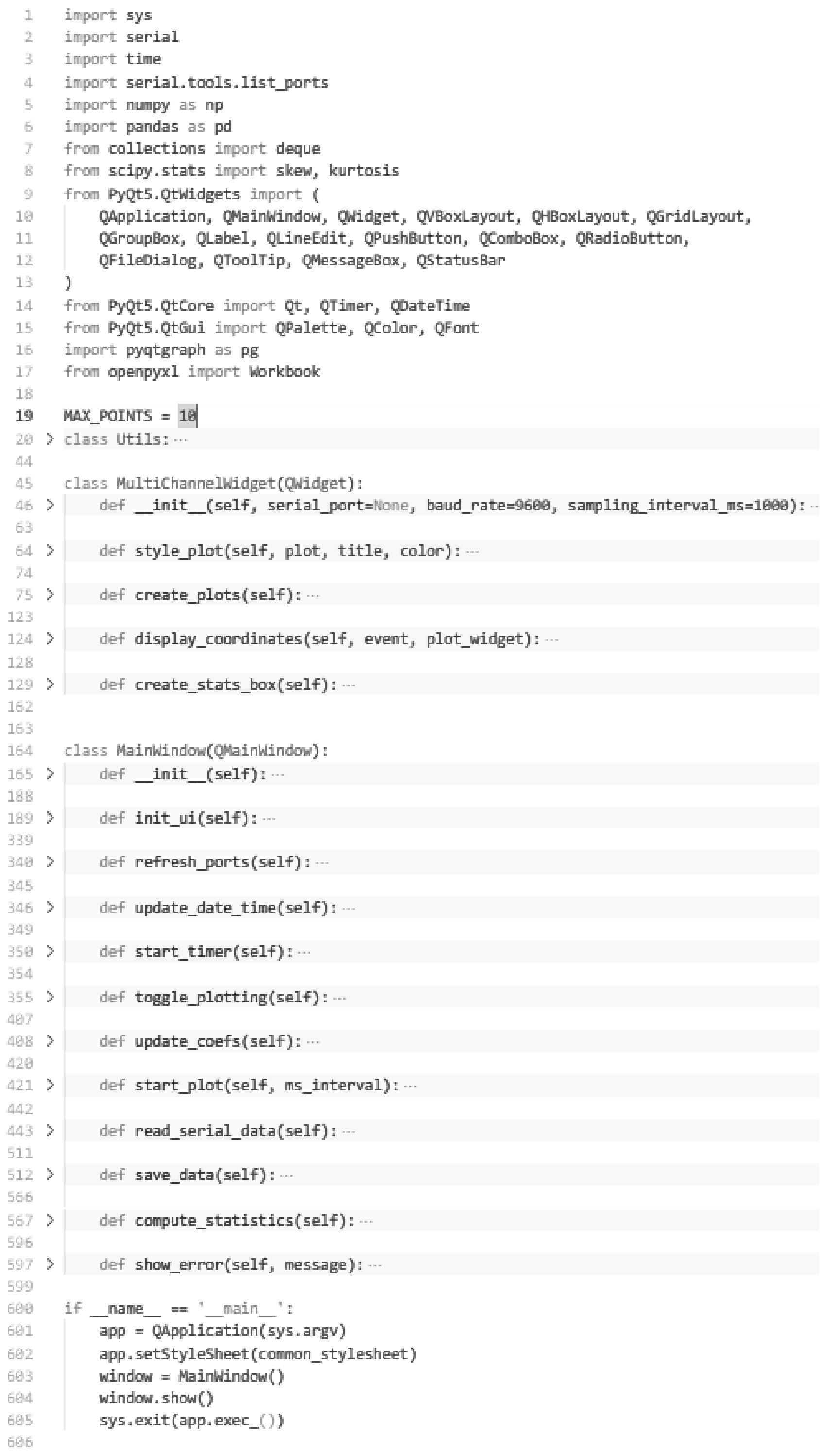
Flow vs Volume plot

It is the most important graph in a spirometer. This graph ideally should be a loop like, a loop of exhalation and inhalation. As the flow increases during exhalation, the volume also increases, starting the positive part of the loop. When the peak [point of exhalation is reached, the pressure difference drops to indicate that you are inhaling. Then it gradually rises to 0 again, from where the loop again starts.

Save data

On clicking this button, a save file dialog box will open. Choose the destination and the file name you want to store the data in. The flow rate and volume will be stored along with its time stamp.

<u>Note:</u> The data will be stored in excel document format



```python
import sys
import serial
import time
import serial.tools.list_ports
import numpy as np
import pandas as pd
from collections import deque
from scipy.stats import skew, kurtosis
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QGridLayout,
    QGroupBox, QLabel, QLineEdit, QPushButton, QComboBox, QRadioButton,
    QFileDialog, QToolTip, QMessageBox, QStatusBar
)
from PyQt5.QtCore import Qt, QTimer, QDateTime
from PyQt5.QtGui import QPalette, QColor, QFont
import pyqtgraph as pg
from openpyxl import Workbook

MAX_POINTS = 10
class Utils: ···

class MultiChannelWidget(QWidget):
    def __init__(self, serial_port=None, baud_rate=9600, sampling_interval_ms=1000): ···

    def style_plot(self, plot, title, color): ···

    def create_plots(self): ···

    def display_coordinates(self, event, plot_widget): ···

    def create_stats_box(self): ···


class MainWindow(QMainWindow):
    def __init__(self): ···

    def init_ui(self): ···

    def refresh_ports(self): ···

    def update_date_time(self): ···

    def start_timer(self): ···

    def toggle_plotting(self): ···

    def update_coefs(self): ···

    def start_plot(self, ms_interval): ···

    def read_serial_data(self): ···

    def save_data(self): ···

    def compute_statistics(self): ···

    def show_error(self, message): ···

if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.setStyleSheet(common_stylesheet)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

Figure 13: GUI functions for Spirometer

6. TESTING OF DEVELOPED EMBEDDED SYSTEMS

6.1 4 channel frequency meter

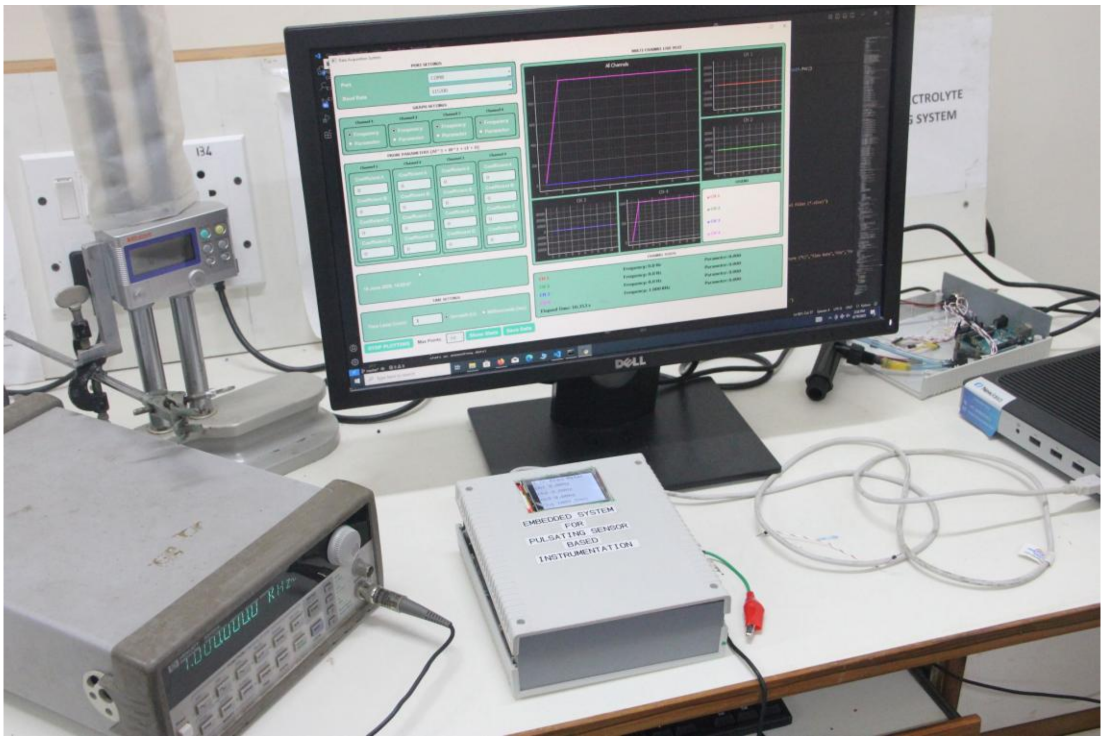
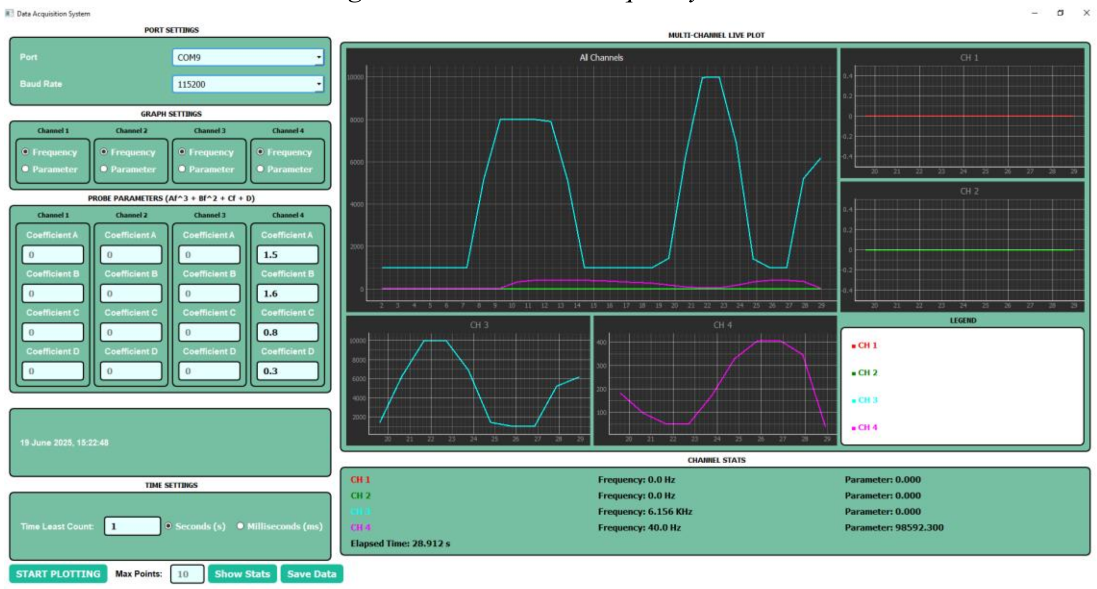


Figure 14: 4 channel Frequency Meter



PORT SETTINGS

Port: COM9
Baud Rate: 115200

GRAPH SETTINGS

Channel 1	Channel 2	Channel 3	Channel 4
[✓ Frequency] Parameter	[✓ Frequency] Parameter	[✓ Frequency] Parameter	[✓ Frequency] Parameter

PROBE PARAMETERS (Af^3 + Bf^2 + Cf + D)

	Channel 1	Channel 2	Channel 3	Channel 4
Coefficient A	0	0	0	1.5
Coefficient B	0	0	0	1.6
Coefficient C	0	0	0	0.8
Coefficient D	0	0	0	0.3

19 June 2025, 15:22:48

TIME SETTINGS

Time Least Count: 1 [✓ Seconds (s)] Milliseconds (ms)

Max Points: 10

CHANNEL STATS

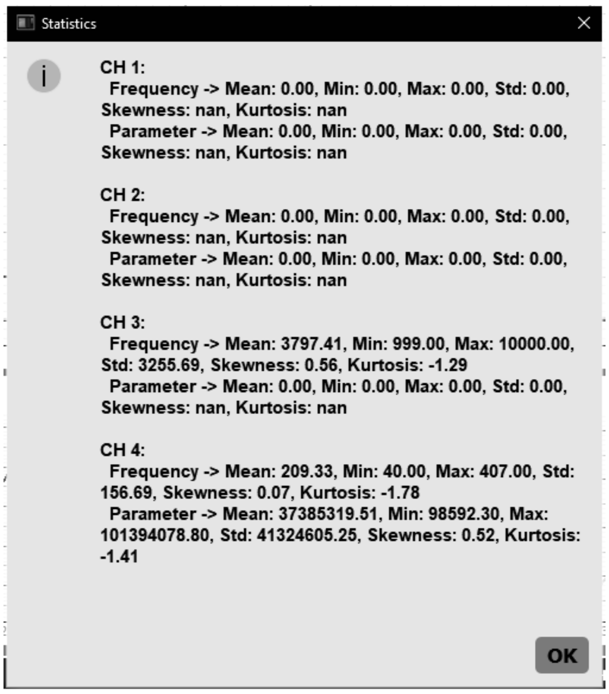
```
CH 1    Frequency: 0.0 Hz       Parameter: 0.000
CH 2    Frequency: 0.0 Hz       Parameter: 0.000
CH 3    Frequency: 6.156 KHz    Parameter: 0.000
CH 4    Frequency: 40.0 Hz      Parameter: 98592.300
Elapsed Time: 28.912 s
```

Figure 15: GUI

Table 12: Channel 3 Log

Time Stamp	Frequency	Parameter
19 June 2025, 15:22:09	1000	0
19 June 2025, 15:22:10	1000	0
19 June 2025, 15:22:11	1000	0
19 June 2025, 15:22:12	1000	0
19 June 2025, 15:22:13	1000	0
19 June 2025, 15:22:14	1000	0
19 June 2025, 15:22:15	5159	0
19 June 2025, 15:22:16	8000	0
19 June 2025, 15:22:17	8000	0
19 June 2025, 15:22:18	8000	0
19 June 2025, 15:22:19	7894	0
19 June 2025, 15:22:20	5103	0
19 June 2025, 15:22:21	1000	0

19 June 2025, 15:22:22	1000	0
19 June 2025, 15:22:24	999	0
19 June 2025, 15:22:25	999	0
19 June 2025, 15:22:26	999	0
19 June 2025, 15:22:27	1445	0
19 June 2025, 15:22:28	6285	0
19 June 2025, 15:22:29	9976	0
19 June 2025, 15:22:30	10000	0
19 June 2025, 15:22:31	6883	0
19 June 2025, 15:22:32	1419	0
19 June 2025, 15:22:33	1000	0
19 June 2025, 15:22:34	1000	0
19 June 2025, 15:22:35	5213	0
19 June 2025, 15:22:36	6156	0



```
CH 1:
  Frequency -> Mean: 0.00, Min: 0.00, Max: 0.00, Std: 0.00,
Skewness: nan, Kurtosis: nan
  Parameter -> Mean: 0.00, Min: 0.00, Max: 0.00, Std: 0.00,
Skewness: nan, Kurtosis: nan

CH 2:
  Frequency -> Mean: 0.00, Min: 0.00, Max: 0.00, Std: 0.00,
Skewness: nan, Kurtosis: nan
  Parameter -> Mean: 0.00, Min: 0.00, Max: 0.00, Std: 0.00,
Skewness: nan, Kurtosis: nan

CH 3:
  Frequency -> Mean: 3797.41, Min: 999.00, Max: 10000.00,
Std: 3255.69, Skewness: 0.56, Kurtosis: -1.29
  Parameter -> Mean: 0.00, Min: 0.00, Max: 0.00, Std: 0.00,
Skewness: nan, Kurtosis: nan

CH 4:
  Frequency -> Mean: 209.33, Min: 40.00, Max: 407.00, Std:
156.69, Skewness: 0.07, Kurtosis: -1.78
  Parameter -> Mean: 37385319.51, Min: 98592.30, Max:
101394078.80, Std: 41324605.25, Skewness: 0.52, Kurtosis:
-1.41
```

Figure 16: GUI statistics

Table 13: Channel 4 Log

Time Stamp	Frequency	Parameter
19 June 2025, 15:22:09	40	98592.3
19 June 2025, 15:22:10	40	98592.3
19 June 2025, 15:22:11	40	98592.3
19 June 2025, 15:22:12	40	98592.3
19 June 2025, 15:22:13	40	98592.3
19 June 2025, 15:22:14	40	98592.3
19 June 2025, 15:22:15	40	98592.3

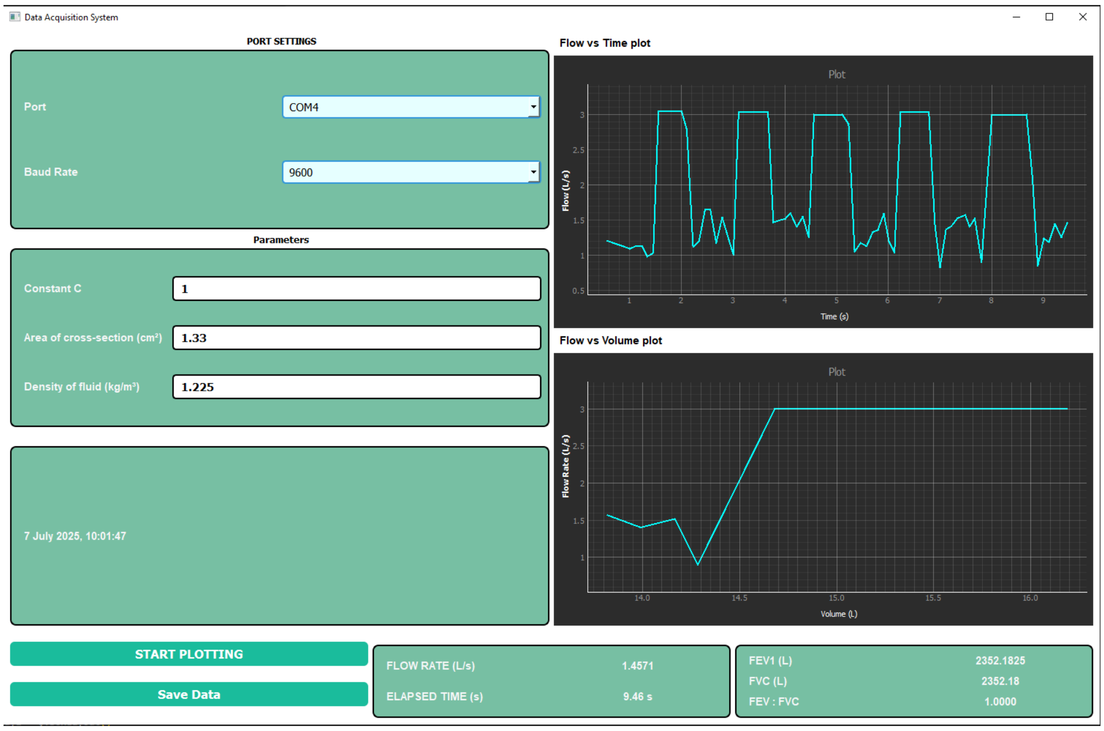
19 June 2025, 15:22:16	40	98592.3
19 June 2025, 15:22:17	326	52139266.7
19 June 2025, 15:22:18	405	99907951.8
19 June 2025, 15:22:19	406	100649186.7
19 June 2025, 15:22:20	406	100649186.7
19 June 2025, 15:22:21	405	99907951.8
19 June 2025, 15:22:22	384	85170893.1
19 June 2025, 15:22:24	342	60189948.3
19 June 2025, 15:22:25	306	43128986.7
19 June 2025, 15:22:26	273	30639090.6
19 June 2025, 15:22:27	181	8947174.2
19 June 2025, 15:22:28	96	1341926.7
19 June 2025, 15:22:29	51	203179.2
19 June 2025, 15:22:30	51	203179.2

19 June 2025, 15:22:31	172	7680144.3
19 June 2025, 15:22:32	329	53590382.6
19 June 2025, 15:22:33	405	99907951.8
19 June 2025, 15:22:34	407	101394078.8
19 June 2025, 15:22:35	347	62865816.8
19 June 2025, 15:22:36	40	98592.3

Table 14: Statistics Log

Channel	Freq Mean	Freq Min	Freq Max	Freq Std	Freq Skew	Freq Kurt	Param Mean	Param Min	Param Max	Param Std	Param Skew	Param Kurt
CH 1	0	0	0	0			0	0	0	0		
CH 2	0	0	0	0			0	0	0	0		
CH 3	3797.41	999	10000	3255.69	0.56	-1.29	0	0	0	0		
CH 4	209.33	40	407	156.69	0.07	-1.78	37385319.5	98592.3	101394079	41324605	0.52	-1.41

6.2 Spirometer



Port: COM4

Baud Rate: 9600

Constant C: 1

Area of cross-section (cm²): 1.33

Density of fluid (kg/m³): 1.225

7 July 2025, 10:01:47

FLOW RATE (L/s)	1.4571
ELAPSED TIME (s)	9.46 s
FEV1 (L)	2352.1825
FVC (L)	2352.18
FEV : FVC	1.0000

Figure 17: Spirometer GUI

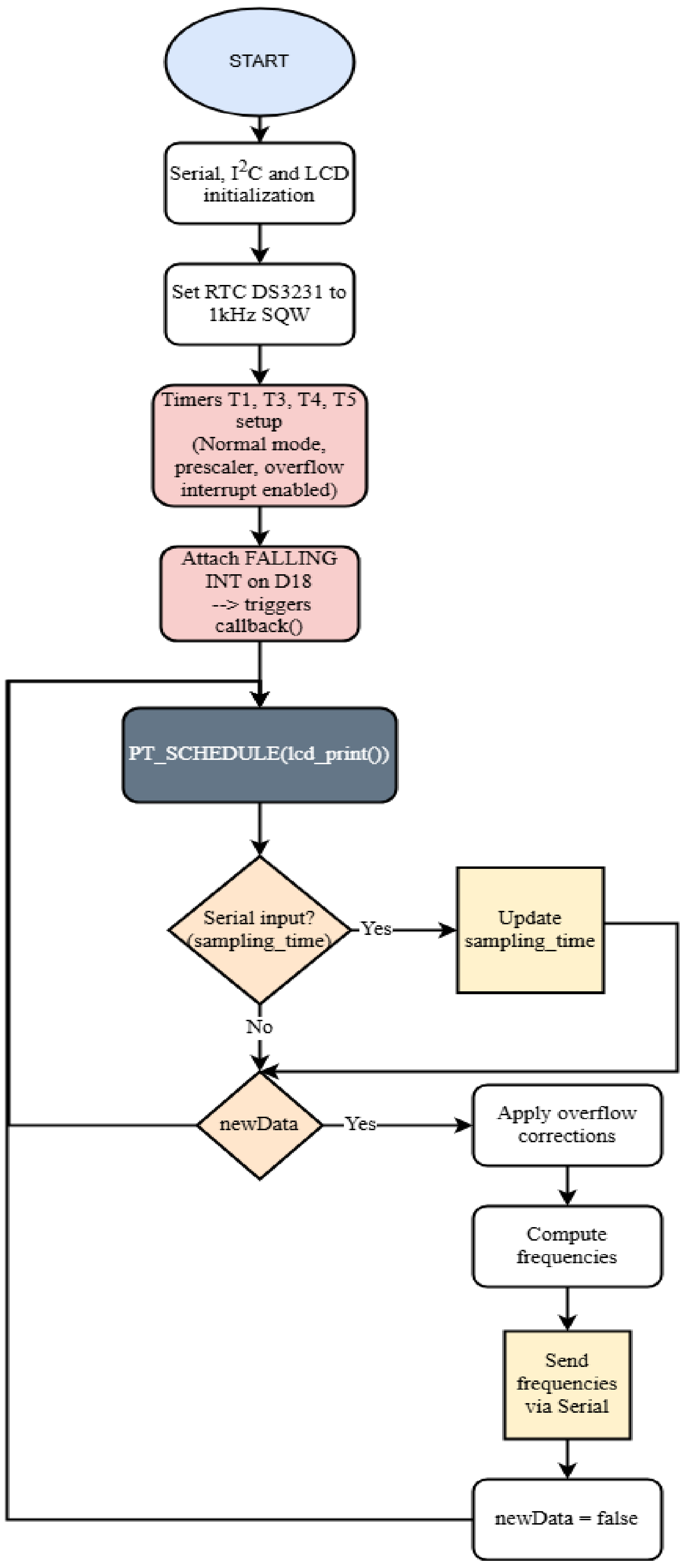
7. CONCLUSION AND FUTURE WORK

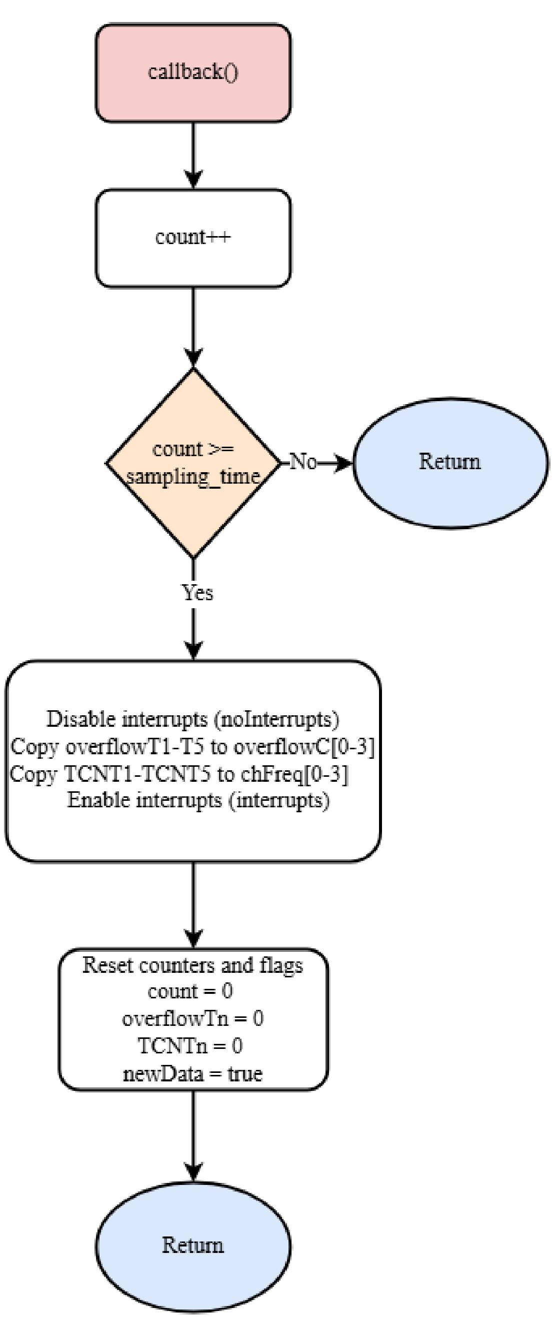
The internship provided valuable insights into both embedded systems and GUI-based software development. Two fully functional GUI tools were successfully implemented—one for frequency-based sensor monitoring and another for spirometry analysis. The pulsating sensor GUI supports real-time multi-channel monitoring, statistical evaluation, and user-defined parameters for converting frequency data into meaningful measurements. The spirometer GUI provides critical respiratory analysis features such as real-time flow visualization and FEV1/FVC computation, aiding in pulmonary health diagnostics.

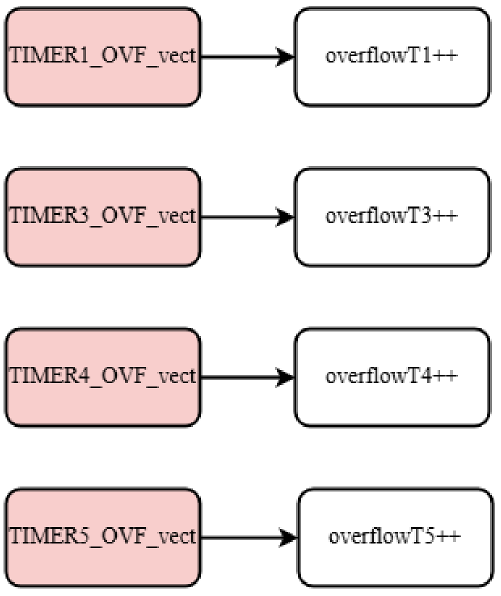
The project demonstrates a holistic application of hardware programming, sensor integration, and user interface development, offering a practical solution for industrial and medical instrumentation. It also emphasized the importance of efficient data handling, user-centric design, and robust communication protocols. The experience gained through this internship lays a strong foundation for future work in embedded health-tech and instrumentation systems.

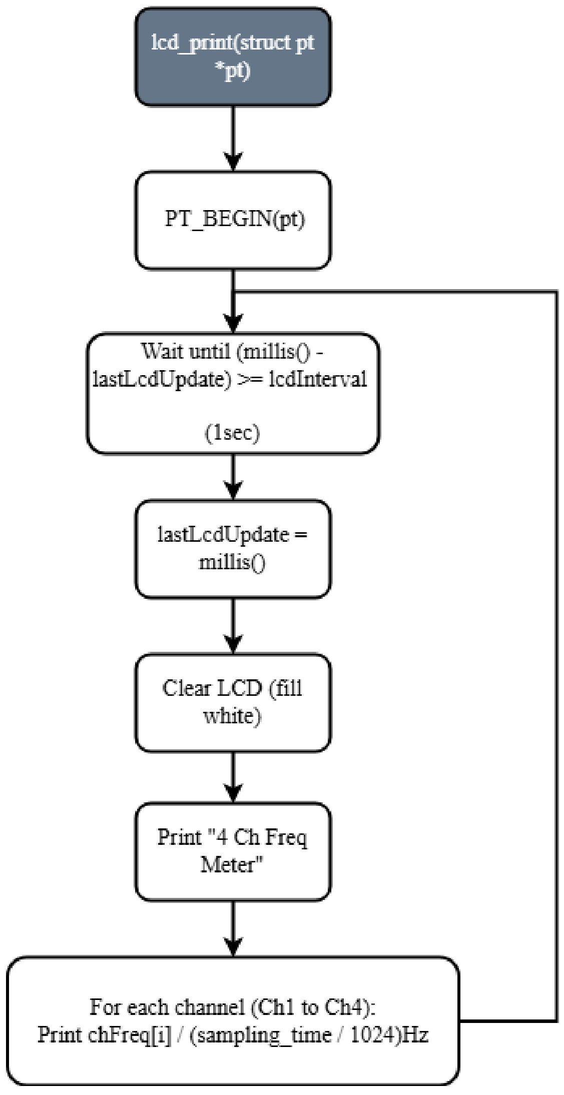
8. APPENDIX - 1 : FLOWCHART FOR ARDUINO CODES

4 - channel frequency meter

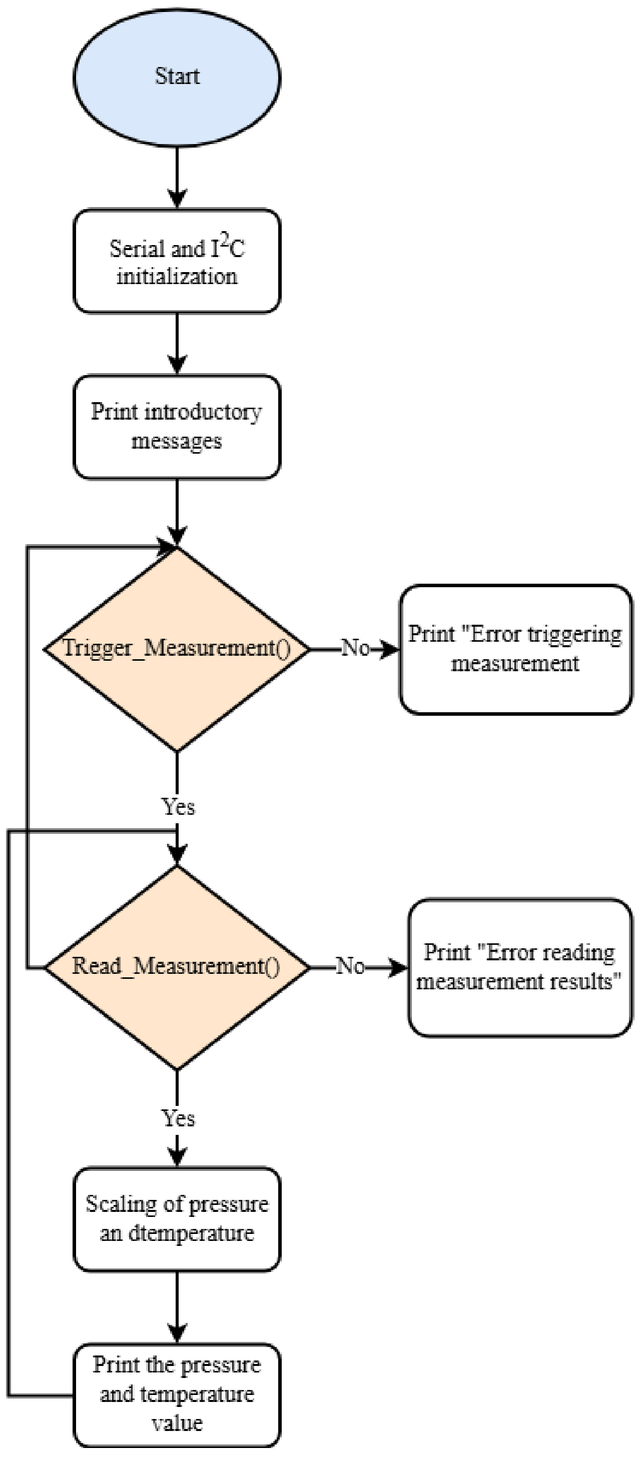

START
Serial, I2C and LCD initialization
Set RTC DS3231 to 1kHz SQW
Timers T1, T3, T4, T5 setup (Normal mode, prescaler, overflow interrupt enabled)
Attach FALLING INT on D18 --> triggers callback()
PT_SCHEDULE(lcd_print())
Serial input? (sampling_time)
Yes
Update sampling_time
No
newData
Yes
Apply overflow corrections
Compute frequencies
Send frequencies via Serial
newData = false


callback()
count++
count >= sampling_time
No
Return
Yes
Disable interrupts (noInterrupts)
Copy overflowT1-T5 to overflowC[0-3]
Copy TCNT1-TCNT5 to chFreq[0-3]
Enable interrupts (interrupts)
Reset counters and flags
count = 0
overflowTn = 0
TCNTn = 0
newData = true
Return

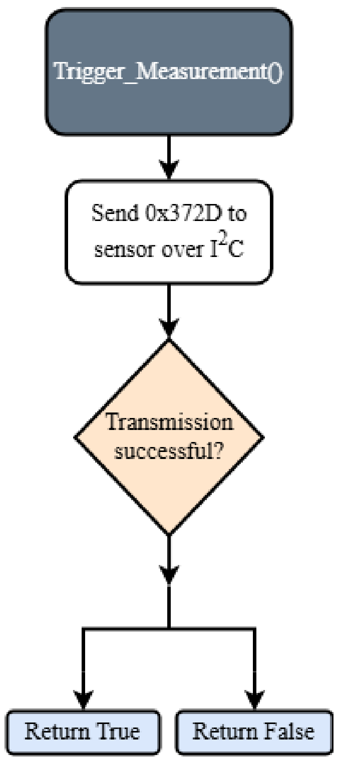

TIMER1_OVF_vect
overflowT1++
TIMER3_OVF_vect
overflowT3++
TIMER4_OVF_vect
overflowT4++
TIMER5_OVF_vect
overflowT5++


lcd_print(struct pt *pt)
PT_BEGIN(pt)
Wait until (millis() - lastLcdUpdate) >= lcdInterval (1sec)
lastLcdUpdate = millis()
Clear LCD (fill white)
Print "4 Ch Freq Meter"
For each channel (Ch1 to Ch4):
Print chFreq[i] / (sampling_time / 1024)Hz

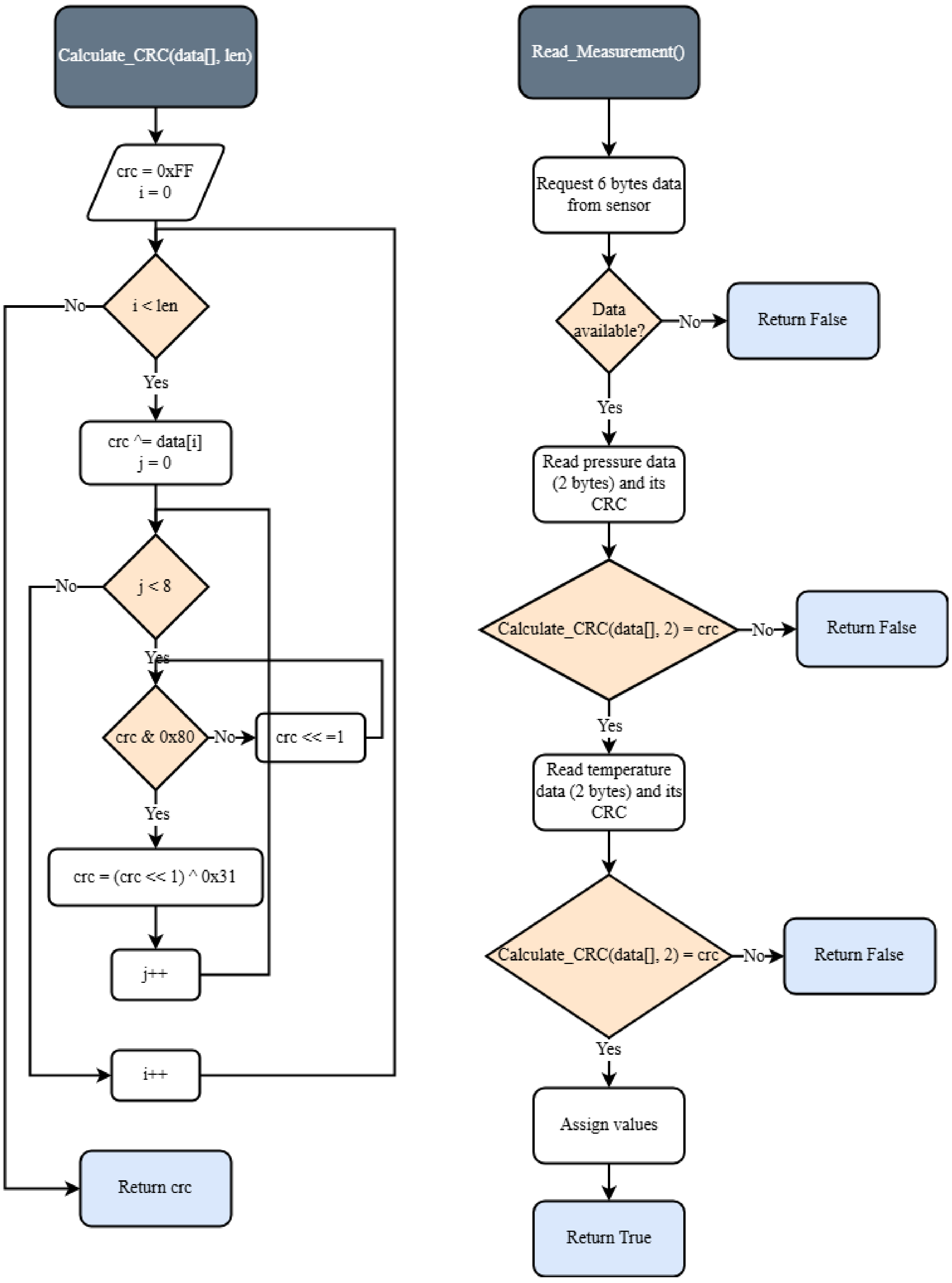
Spirometer



Start
Serial and I^{2}C initialization
Print introductory messages
Trigger_Measurement()
No
Print "Error triggering measurement
Yes
Read_Measurement()
No
Print "Error reading measurement results"
Yes
Scaling of pressure an dtemperature
Print the pressure and temperature value



Trigger_Measurement()
Send 0x372D to sensor over I^{2}C
Transmission successful?
Return True
Return False


Calculate_CRC(data[], len)
crc = 0xFF
i = 0
i < len
No
Yes
crc ^= data[i]
j = 0
j < 8
No
Yes
crc & 0x80
No
crc << =1
Yes
crc = (crc << 1) ^ 0x31
j++
i++
Return crc
Read_Measurement()
Request 6 bytes data from sensor
Data available?
No
Return False
Yes
Read pressure data (2 bytes) and its CRC
Calculate_CRC(data[], 2) = crc
No
Return False
Yes
Read temperature data (2 bytes) and its CRC
Calculate_CRC(data[], 2) = crc
No
Return False
Yes
Assign values
Return True

REFERENCES

[1] “IGCAR Profile.” Accessed: July 26, 2025. [Online]. Available: https://www.igcar.gov.in/igcarprofile.html

[2] “Introduction — PyQt Documentation v5.15.7.” Accessed: July 26, 2025. [Online]. Available: https://www.riverbankcomputing.com/static/Docs/PyQt5/introduction.html

[3] “ATmega640/1280/1281/2560/2561 datasheet.” Accessed: July 26, 2025. [Online]. Available: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[4] S. Santos, “Arduino DS3231 Real Time Clock (RTC): Time and Alarms | Random Nerd Tutorials.” Accessed: July 26, 2025. [Online]. Available: https://randomnerdtutorials.com/arduino-ds3231-real-time-clock/

[5] “2.4inch Arduino Display - LCD wiki.” Accessed: July 26, 2025. [Online]. Available: https://www.lcdwiki.com/2.4inch_Arduino_Display

[6] S. Santos, “Guide to SD Card Module with Arduino | Random Nerd Tutorials.” Accessed: July 26, 2025. [Online]. Available: https://randomnerdtutorials.com/guide-to-sd-card-module-with-arduino/

[7] “arduino-mega2560-schematic.pdf.” Accessed: July 26, 2025. [Online]. Available: https://www.arduino.cc/en/uploads/Main/arduino-mega2560-schematic.pdf

[8] “Spirometry,” *Wikipedia*. July 17, 2025. Accessed: July 26, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Spirometry&oldid=1301028651